Reconnaissance d'objets 3D à l'aide d'arbres de classification*

Bruno Jedynak <bruno.jedynak@inria.fr>
François Fleuret <francois.fleuret@inria.fr>

9 avril 1996

Résumé

Notre but est de développer un algorithme pour la reconnaissance d'objets 3D à partir d'images. Nous proposons une approche empirique basée sur un apprentissage à partir de bases de données d'images, constituées de nombreuses vues de chaque objet dans différentes positions. Nous n'utilisons pas de modèles pour la géométrie des objets ou la photométrie de la scène. Comme dans [7], [9], nous proposons d'utiliser des arbres de classification ([3]); la méthodologie globale pour notre travail provient de [1].

L'idée de base est la suivante : tout d'abord associer à chaque pixel de l'image un code décrivant la topographie du voisinage (uniforme, bord, coin, texture, etc.) ; puis, poser de nombreuses questions sur la disposition spatiale de ces codes. Par exemple, nous pourrions demander "y a-t-il un pixel de code T_1 au nord d'un autre de code T_2 et au sud d'un troisième de code T_3 ?". La réponse est "oui" ou "non", selon que la configuration cherchée apparaît ou pas dans l'image. Comme les configurations ne font intervenir que des contraintes relatives sur les angles, les questions associées jouissent de bonnes propriétés d'invariance.

1 Introduction

Notre but est de développer un algorithme pour la reconnaissance, à partir d'images bitmap, d'objets 3D présents dans des scènes complexes. Nous supposons donnée une petite liste d'objets (de 10 à 100 objets) et nous souhaitons détecter et identifier toutes les instances de ces objets. Nous n'essayerons pas de réaliser une reconstruction 3D de la scène, même si des informations qualitatives sur la position des objets détectés pourront être disponibles *après* la reconnaissance.

Bien que la reconnaissance d'objets dans une scène complète soit notre but final, nous ne considérons ici que le cas ou un objet seul occupe une partie significative de l'image. De notre point de vue, résoudre ce problème est un premier pas dans la direction de l'analyse de scènes complètes.

Nous proposons une approche empirique basée sur un apprentissage à partir d'une bases de données d'images. Nous proposons de caractériser l'ensemble des images de la base de données d'apprentissage à l'aide d'une famille de graphes orientés "superposables" sur les images. À un sommet d'un tel graphe est associée une fonction booléenne, appelée code. Un sommet est superposable sur un pixel d'une image si la fonction booléenne évaluée en ce pixel prend la valeur 1. Cette fonction dépend du niveau de gris de ce pixel ainsi que des niveaux de gris des pixels voisins. Les codes sont construits de telle sorte que deux pixels ont le même code si les topographies de leurs voisinages sont similaires (uniforme, bord, coin,

*. A paraître dans les actes d'Image'com 96, Bordeaux, FRANCE, 20-22 Mai 1996

texture, etc.), voir paragraphe 3. Les arêtes du graphe correspondent à des relations géométriques entre codes, comme par exemple “au Nord de” ou encore “au Sud Est de”.

Chaque graphe permet de séparer en deux l’ensemble des images de la base d’apprentissage : les images sur lesquelles ce graphe est superposable, et les autres. Itérativement, le partage peut être raffiné donnant naissance à une arborescence, ou *arbre de classification*.

L’utilisation d’arbres de classification (voir [3], [13], [14], [5], [6], [16]) pour des problèmes de reconnaissance de formes n’est pas nouvelle. Cette technique est utilisée pour la classification de formes rigides 2D dans [1], [2], [4], [23], [21] et [18]. Dans [10], [11], [8] [21] [23], les auteurs considèrent des objets 2D déformables et dans [19], [20], [22] des objets 3D rigides. Notre travail est une continuation de [1], où les auteurs s’intéressent à la reconnaissance d’objets 2D déformables dans des images binaires (caractères manuscrits). Nous proposons d’utiliser la même technique pour des objets 3D et des images en niveaux de gris. Avec [1], nous nous différencions de l’ensemble des publications citées dans ce paragraphe par l’utilisation de graphes connexes et complexes afin de caractériser les objets (voir paragraphe 4). Nous pensons ainsi obtenir une caractérisation de nature topologique, suffisamment riche et souple pour être invariante aux changements d’aspect.

Au paragraphe 2 nous présentons la base de données sur laquelle nous évaluons nos résultats. Nous avons choisi une base de données pour laquelle des algorithmes de natures différentes au notre ont été testés ([12],[15] et[17]). Au paragraphe 3, nous présentons le recodage des images. Au paragraphe 4, nous présentons la classification elle-même. Au paragraphe 5 nous décrivons nos résultats, et enfin le paragraphe 6 présente nos conclusions et perspectives.

2 Bases de données

Nous illustrons le fonctionnement et les résultats de notre algorithme sur les images de la “Columbia Object Image Library (COIL-20) [1]”. Cette base de données contient 1440 images 128x128 codées sur 8 bits de 20 objets sur fond noir (72 images de chaque objet). Les objets sont disposés tour à tour sur une table tournante. Une caméra fixe acquiert une image tous les 5 degrés d’angle. Les objets sont normalisés en dimensions de façon à remplir au mieux l’image, sans pour autant les déformer. De plus les images sont normalisées en intensité. Les objets sont présentés sur la figure 1. Le protocole d’utilisation de cette base de données, utilisé dans [12],[15] et[17], consiste à extraire une image sur deux (par exemple les images de numéro pair) pour l’apprentissage, et à effectuer le test, et donc à évaluer les résultats de l’algorithme sur l’autre moitié des images. Ces auteurs présentent d’excellents taux de reconnaissance avec ce protocole.

Calculons, pour une image quelconque u de l’ensemble de test, l’image v de l’ensemble d’apprentissage la plus proche au sens de la norme L^2. Nous remarquons alors que u et v sont *toujours* des images d’un même objet. Cet algorithme très simple fournit donc un taux de reconnaissance de 100%. En revanche, il est coûteux en temps de calcul et en mémoire puisqu’il faut disposer de toutes les images de la base d’apprentissage lors de la reconnaissance. Dans [12], les auteurs présentent un algorithme qui permet de calculer de manière efficace une distance entre images qui approche la norme L^2. Au vu du résultat présenté ci-dessus, cette approche nous semble ici la plus légitime. Toutefois, pour des problèmes plus difficiles, faisant

1. Cette base de données d’images a été réalisée par Sameer A. Nene, Shree K. Nayar et Hiroshi Murase. Elle est disponible à des fins de recherche sur le site http://www.cs.columbia.edu/CAVE/visual-learn-recog.html

FIG. 1 - *Les tests ont été faits sur les images de la "Columbia Object Image Library (COIL-20)"*

intervenir un fond complexe, du bruit ou des occlusions, nous pensons qu'elle sera peu robuste.

Dans [15] et[17], les auteurs caractérisent une image à l'aide d'un petit ensemble de pixels (petit par rapport au nombre total de pixels dans une image). A chacun est associé un ensemble de nombres, ou vecteur, décrivant de manière fine la topographie du voisinage de ce pixel. L'ensemble de ces vecteurs est un index pour une image et jouit de propriétés d'invariance relatives aux déplacements de l'objet. La classification d'une nouvelle image s'effectue en comparant de manière appropriée son index avec ceux des images d'apprentissage.

3 Recodage

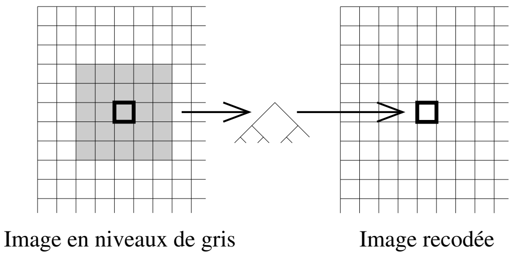

FIG. 2 - *Le recodage consiste à construire une image dans laquelle chaque pixel porte un code décrivant son voisinage 5×5 dans l'image originale. Ce code est obtenu à l'aide d'un arbre de classification.*

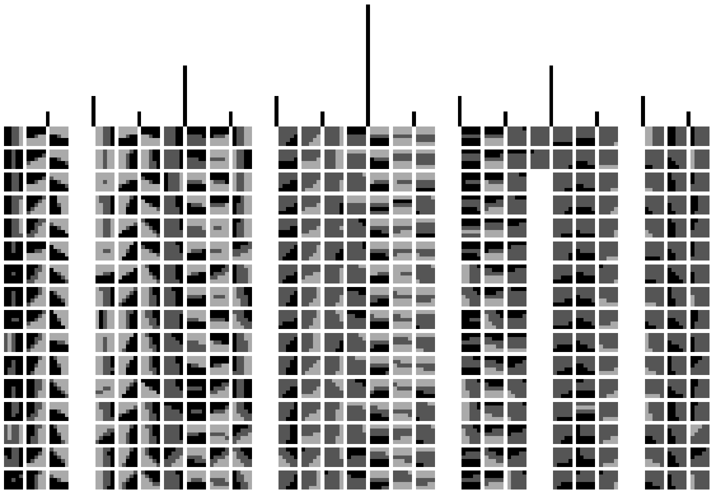

FIG. 3 - *A chaque imagette* 5×5 *est associé un code entier compris entre 0 et 31. Chaque colonne de cette image contient des exemples d'imagettes ayant le même code (il y a donc 32 colonnes). Les imagettes d'un même groupe ont des propriétés géométriques communes.*

La première étape de notre algorithme consiste à recoder les images en "étiquettes", c'est à dire à remplacer en chaque pixel la donnée de niveau de gris par un code entier qui décrit la topographie de l'imagette 5×5 centrée sur le pixel (cf. figure 2). Des exemples d'images ainsi recodées sont présentés figure 4.

3.1 Calcul du code d'une imagette

Le code associé à une imagette est calculé à l'aide d'un arbre de classification. Il s'agit de quantification vectorielle. Chaque nœud (non terminal) de cet arbre possède deux fils et porte une question qui compare la valeur de gris d'un des pixels de l'imagette avec celle du pixel central. Cette comparaison peut être de la forme "le pixel est-il plus clair que le centre", "plus foncé" ou "de même gris" (pour une tolérance de $1/16$ de l'échelle de gris). En fonction de la réponse à la question, l'image tombe à un autre nœud, où une autre question est posée, etc. Après avoir posé 5 questions, nous obtenons un code de 5 bits composé des réponses aux questions. Le nombres d'opérations nécessaires au recodage est donc de 5 comparaisons par pixel. Sur la figure 3 sont représentés des exemples d'imagettes ayant le même code.

3.2 Construction de l'arbre de classification pour les imagettes

L'arbre utilisé pour le calcul des codes des imagettes est construit de manière à répartir le plus uniformément possible les imagettes dans les feuilles terminales. Nous le construirons de la manière suivante : nous commençons par extraire aléatoirement plusieurs dizaines de milliers d'imagettes 5×5 des images de la base d'apprentissage. Pour construire un nœud, nous déterminons la meilleure question à y placer.

Nous disposons de trois types de questions : "le pixel est-il plus clair que le centre", "plus foncé" ou "de même gris", et comme chaque question teste un pixel de l'imagette, il y en a 25 de chaque type, soit finalement 75 questions.

Chaque question partage l'ensemble des imagettes qui arrivent à un nœud en deux parties (le sous-ensemble des imagettes qui répondent oui, et celui des ima-

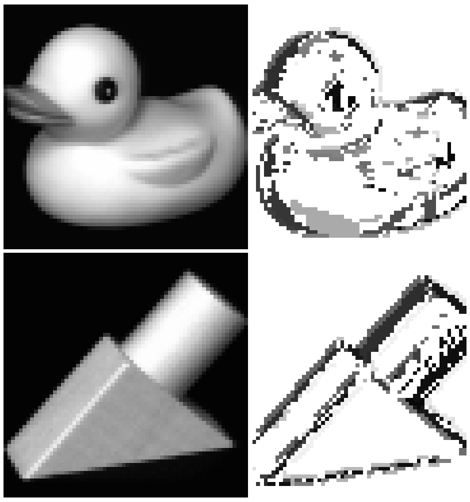

FIG. 4 - *La première étape de l'algorithme est un recodage des images, qui permet d'obtenir en chaque point un code décrivant le voisinage.*

gettes qui répondent non). Nous choisissons de placer au nœud la question qui fait le partage le plus équilibré.

4 Reconnaissance

4.1 Arbre de classification

La reconnaissance des objets à partir des images recodées se fait à l'aide d'un autre arbre de classification. Chaque sommet intérieur de cet arbre (ou nœud) teste la présence dans l'image d'une configuration géométrique entre deux codes. Il possède deux nœuds fils, un pour "la configuration est présente", l'autre pour "la configuration n'est pas présente". Chaque feuille porte le résultat de la classification pour les images qui y parviennent; ce résultat est une distribution de probabilité sur les objets.

Dans le programme qui a été développé pour les expérimentations présentées dans cet article, les configurations sont paramétrées par deux codes T_1 et T_2, et deux angles α et β. La configuration définie par ces paramètres est présente dans l'image s'il existe deux pixels, l'un portant un code T_1, l'autre un code T_2, et tels que le vecteur qu'ils définissent dans le plan image fasse un angle avec l'horizontale compris entre $\alpha - \beta$ et $\alpha + \beta$ (cf. figure 5).

Hormis le premier nœud de l'arbre, tous les nœuds testent une configuration qui fait intervenir un pixel déjà utilisé dans une configuration d'un nœud précédemment rencontré.

Dans le cas où la configuration est présente dans l'image, nous pouvons lui associer un graphe dont les sommets sont localisés sur les pixels intervenant dans la configuration. Comme la configuration recherchée à chaque nœud fait intervenir un pixel présent dans la configuration d'un nœud précédent, ce graphe est connexe (et comme chaque configuration ne fait intervenir que deux pixels, dont un nouveau, il est également sans cycle). Le nombre de sommets de ce graphe dépend du nombre de réponses positives dans les nœuds précédents et de la profondeur du nœud. Une image qui répond "oui" à chaque nœud porte un graphe qui a autant de sommets que de nœud visités plus un (puisque le premier nœud fait intervenir deux nouveaux pixels); alors qu'une image qui répond toujours non porte un graphe vide.

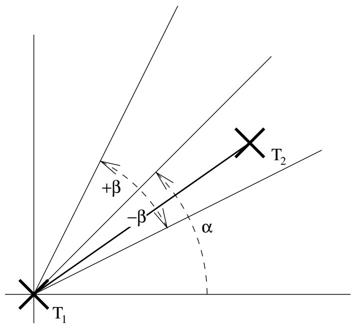

FIG. 5 - *Les relations recherchées à chaque nœud sont de la forme "existe-t-il un pixel portant le code T_1 et un pixel portant le code T_2 tels que le segment qu'ils définissent fasse un angle compris entre $\alpha - \beta$ et $\alpha + \beta$ avec l'horizontale"*

Remarquons que les configurations recherchées par les nœuds de l'arbre peuvent être trouvées en plusieurs endroits de l'image (cf. figure 6).

4.2 Construction des arbres

A la différence de la construction de l'arbre de classification utilisé pour le recodage, il s'agit ici d'apprentissage supervisé. Nous savons quelle est la classe de chacune des images de la base d'apprentissage (il y a une classe par objet).

La construction d'un arbre se fait de manière récursive. Pour construire un sommet, nous commençons par calculer la distribution empirique sur les objets, à l'aide des images de la base d'apprentissage qui y arrivent. Si cette distribution est déterministe, alors ce sommet sera une feuille (dans le cadre de cet article, les distributions résultats portées par les feuilles sont donc déterministes). Dans le cas contraire, le sommet sera un nœud interne, et il faut déterminer la configuration qu'il doit chercher. Ce choix se fera dans un sous-ensemble de configurations candidates, en prenant celle qui diminue le plus l'entropie de Shannon de la distribution empirique sur les objets. Ce sous-ensemble contient quelques centaines de configurations et est généré aléatoirement.

4.3 Utilisation de plusieurs arbres

Lors de la construction des arbres, nous avons utilisé à chaque nœud un sous-ensemble de questions généré de manière aléatoire. Si nous utilisons des sous-ensembles différents, nous obtiendrons des arbres différents. Pour obtenir une robustesse supérieure, nous utilisons plusieurs dizaines d'arbres ainsi construits dont nous combinons les résultats. Pour classifier une image, nous déterminons pour chacun des arbres à quelle feuille tombe l'image, et quelle est la distribution sur les objets associée. La distribution finale est obtenue en calculant la moyenne de toutes ces distributions. Voir à ce propos la discussion dans [1].

5 Résultats

Un test effectué avec 140 arbres, construits à l'aide de la moitié des images de la base de données (soit 36 images de chacun des 20 objets) a donné un taux de reconnaissance de 98.9% sur l'autre moitié des images. Le temps de reconnaissance est de 3267 secondes sur une SPARCStation 20, soit 4.5 secondes par image. L'essentiel de l'erreur provient de la confusion entre des objets similaires (les voitures,

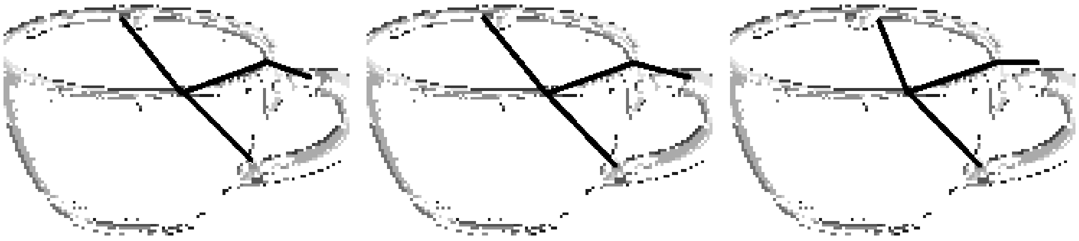

FIG. 6 - *A chaque feuille d'un arbre de classification correspond un graphe qui peut être présent en plusieurs endroits sur l'image. On voit ici les différentes instances d'un même graphe sur l'image d'une tasse. Les sommets de ce graphe sont positionnés sur des pixels. Les arêtes indiquent quels sont les couples de pixels liés par une contrainte géométrique.*

et les boites, cf. figure 1). L'occupation totale en mémoire pour stocker les arbres est de 2.5Mo.

6 Conclusion et perspectives

Nous avons présenté un algorithme générique pour la reconnaissance d'objets dans des images. Le mécanisme de reconnaissance est comme suit : tout d'abord associer à chaque pixel de l'image un code décrivant la topographie du voisinage (uniforme, bord, coin, texture, etc.) ; puis, poser de nombreuses questions sur la disposition spatiale de ces codes. Cet algorithme, évalué sur la base de données "Columbia Object Image Library (COIL-20)" présente de bonnes performances en taux de reconnaissance. Le temps de calcul pour la reconnaissance est de l'ordre de 4.5 seconde par image, mesuré sur une station SPARCStation 20. Toute l'information acquise lors de l'apprentissage est résumée dans les arbres de classification représentant un volume de données total de 2.5Mo.

Notre ambition est de reconnaître des objets dans leur environnement. Nous nous intéressons actuellement à des images de matériel de bureau : agrafeuse, boite, stylo, ... posé sur une table ou un bureau éventuellement texturé, et où d'autres objets : feuilles, téléphone, etc ... sont présents. Dans ces conditions, il est très difficile de segmenter les objets et la variabilité du contexte rend les algorithmes de comparaison d'images pixels à pixel inopérante.

Dans cette perspective et en accord avec [15] et [17], nous préférons utiliser une caractérisation structurelle des pixels (topographie du voisinage) plutôt que d'utiliser directement les niveaux de gris. En accord avec [12] et [17], aucun pixel n'est privilégié a priori. Dans [17], l'utilisation de contraintes géométriques a permis d'élever le taux de reconnaissance de 93.47% à 99.86%. Nous pensons que l'utilisation de relations géométriques entre des sites de l'image caractérisés par leur aspect local devrait permettre de reconnaître des objets de manière robuste au contexte.

Références

[1] Y. Amit, D. Geman, and K. Wilder. Recognizing shapes from simple queries about geometry. July 1995.

[2] E. Arkin, H. Meijer, J. Mitchell, D. Rappaport, and S. Skiena. Decision trees for geometric models. Proc. Ninth ACM Symp. on Computational Geometry, 1993.

[3] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont CA., 1984.

[4] D. Brown, V. Corruble, and C. L. Pittard. A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems. *Pattern Recognition*, 26:953–961, 1993.

[5] W. Buntine and T. Niblett. A further comparison of splitting rules for decision-tree induction. *Machine Learning*, 8:75–85, 1992.

[6] L. A. Clark and D. Pergibon. Tree-based models. In J. M. Chambers and T. J. Hastie, editors, *Statistical Models in S.* Wadsworth & Brooks, Pacific Grove, CA, 1992.

[7] D. Geman and B. Jedynak. Shape recognition using twenty questions. Technical report, INRIA, rapport de recherche 2155, 1993.

[8] D. Geman and B. Jedynak. An active testing model for tracking roads from satellite images. *IEEE Trans. PAMI*, 18:1–15, 1996.

[9] Donald Geman and Bruno Jedynak. An active testing model for tracking roads from satellite images. *IEEE Trans. PAMI (to appear)*, Aout 1994.

[10] Y. X. Gu, Q.R. Wang, and C.Y. Suen. Application of multilayer decision tree in computer recognition of chinese characters. *IEEE Trans. PAMI*, 5:83–89, 1983.

[11] Y.K. Lin and K.S. Fu. Automatic classification of cervical cells using a binary tree classifier. *Pattern Recognition*, 16:69–80, 1983.

[12] H. Murase and S.K. Nayar. Visual learning and recognition of 3d objects from appearance. *International journal of computer vision*, 14:5–24, 1995.

[13] J.R. Quinlan. Induction of decision trees. *MachineLearning*, 1:81–106, 1986.

[14] J.R. Quinlan and R.L. Rivest. Inferring decision trees using minimum description length principle. *Information and Computation*, 80:227–248, 1989.

[15] R.P.N Rao and D.H. Ballard. Object indexing using a iconic sparse distributed memory. In *Proceedings of the 5th international Conference on Computer Vision, Cambridge, Massachusetts, USA*, pages 24–31, 1995.

[16] B. D. Ripley. Neural networks and related methods for classification. *J. Royal Statist. Soc., B*, 56:409–437, 1994.

[17] C. Schmid and R. Mohr. Combining greyvalue invariants with local constraints for object recognition. Submitted to a conference, 1995.

[18] S. Shlien. Multiple binary decision tree classifiers. *Pattern Recognition*, 23:757–763, 1990.

[19] H. Sossa and R. Horaud. Model-indexing: the graph-hashing approach. In *IEEE Conf. Comp. Vision Patt. Recog. Champaign, Ill.*, 1992.

[20] L. Spirkovska. Three-dimensional object recognition using similar triangles and decision trees. *Pattern Recognition*, 26:727–732, 1993.

[21] C.Y. Suen, C. Nadal, R. Legault, T.A. Mai, and L. Lam. Computer recognition of unconstrained handwritten numerals. In *Proc. IEEE*, volume 80, pages 1162–1180, 1992.

[22] M. Swain. Object recognition from a large database using a decision tree. In *Proc. of the DARPA Image Understanding Workshop*. Morgan Kaufman, 1988.

[23] Q.R. Wang and C.Y. Suen. Analysis and design of a decision tree based on entropy reduction and its application to large character set recognition. *IEEE Trans. PAMI*, 6:406–417, 1984.